

Vehicle Speed Detection Using Opencv for Intelligent Transportation Systems

Mrs. S.Subhashini, M.E. Students

Department of Computer Science and Engineering, MRK Institute of Technology, Kattumannar Koil, India

Mrs. T.Kalaiyarasi, Assistant Professor

Department of Computer Science and Engineering, MRK Institute of Technology, Kattumannar Koil India

Mr. A.Akilan, HOD

Department of Computer Science and Engineering, MRK Institute of Technology, Kattumannar Koil India

Abstract

The rapid increase in the number of vehicles has created major challenges such as traffic congestion, road accidents, and traffic management issues. Monitoring vehicle speed is an important part of Intelligent Transportation Systems (ITS) because overspeeding is one of the main causes of accidents. Traditional speed detection methods, such as radar guns and sensors, provide accurate results but are expensive to install and maintain. This research proposes a Vehicle Speed Detection System using Python and OpenCV for real-time traffic monitoring. The system captures traffic videos from surveillance cameras, detects moving vehicles, tracks their movement, and calculates their speed using image processing techniques. Camera calibration is used to improve the accuracy of speed estimation by converting pixel measurements into real-world distances. The proposed system consists of video acquisition, preprocessing, vehicle detection, tracking, speed estimation, and visualization modules. Experimental results show that the system achieved approximately 93% vehicle detection accuracy, 90–95% tracking accuracy, and 88–92% speed estimation accuracy. The framework provides a low-cost, efficient, and scalable solution for traffic monitoring, law enforcement, and smart city applications. Future improvements may include deep learning-based vehicle detection and automated camera calibration techniques.

Keywords: Vehicle Speed Detection, OpenCV, Computer Vision, Intelligent Transportation System, Vehicle Tracking, Traffic Monitoring, Speed Estimation, Image Processing.

I. INTRODUCTION

A. Background

Transportation systems play a vital role in economic development and social mobility. Rapid urbanization and industrial growth have increased the number of vehicles operating on road networks, creating significant challenges in traffic management. Traffic congestion, increased travel time, fuel consumption, environmental pollution, and road accidents are major concerns faced by transportation authorities worldwide.

Among these challenges, vehicle overspeeding is a primary factor contributing to road accidents and fatalities. According to international road safety reports, excessive speed significantly increases both accident probability and accident severity. Consequently, accurate vehicle speed monitoring has become an essential requirement for modern traffic management systems.

Traditional speed monitoring technologies include radar-based systems, laser speed guns, inductive loop detectors, and infrared sensors. While these technologies provide reliable measurements, they involve substantial

installation and maintenance costs. Additionally, their coverage area is limited and scaling them across large transportation networks becomes economically challenging.

Recent advancements in computer vision, image processing, and artificial intelligence have enabled video-based traffic monitoring systems that utilize existing surveillance infrastructure. These systems analyze traffic videos captured by CCTV cameras to identify vehicles, track movement patterns, and estimate vehicle speed without requiring specialized hardware.

B. Intelligent Transportation Systems

Intelligent Transportation Systems (ITS) combine communication technologies, computer systems, sensors, and transportation infrastructure to improve the efficiency, safety, and sustainability of road transportation. ITS provides real-time traffic information, helps manage traffic congestion, optimizes travel routes, and supports accident prevention. In recent years, computer vision has become an important component of ITS because it enables the automatic

analysis of traffic videos and extraction of useful information. Vision-based systems can perform various tasks such as vehicle detection, vehicle classification, traffic density estimation, lane detection, license plate recognition, traffic violation monitoring, and vehicle speed estimation. These capabilities make computer vision a cost-effective and efficient alternative to traditional traffic monitoring systems.

C. Motivation for the Research

Most urban areas already possess extensive CCTV surveillance infrastructure. Utilizing these existing cameras for speed detection eliminates the need for deploying dedicated hardware devices. Furthermore, computer vision-based solutions can simultaneously monitor multiple vehicles, reducing operational costs while improving monitoring coverage.

The motivation behind this research is to develop a low-cost, scalable, and accurate vehicle speed detection framework using OpenCV and Python. The proposed system aims to provide real-time speed monitoring capabilities that can be integrated into smart transportation ecosystems.

D. Problem Statement

Current vehicle speed monitoring systems face several challenges, including high installation and maintenance costs, limited monitoring coverage, dependence on specialized hardware devices, and difficulties in large-scale deployment. In addition, their performance can be affected by environmental conditions such as rain, fog, and poor visibility. Therefore, there is a need for an automated and cost-effective vision-based system that can accurately estimate vehicle speed using standard surveillance cameras and image processing techniques. Such a system can provide efficient traffic monitoring while reducing infrastructure and operational costs.

E. Research Objectives

The primary objective of this study is to develop an efficient Vehicle Speed Detection System using OpenCV and computer vision techniques. The system aims to detect vehicles from traffic videos, track their movement across consecutive frames, and estimate their speed using displacement and time information. The study also focuses on implementing camera calibration techniques to improve measurement accuracy and evaluating system performance under different traffic conditions. Furthermore, the proposed framework seeks to provide a scalable, cost-effective, and reliable solution for intelligent transportation and traffic monitoring applications.

F. Scope of the Study

The proposed Vehicle Speed Detection System focuses on processing recorded traffic videos to detect moving vehicles, track their movement, and estimate their speed in real time. The system uses computer vision techniques and OpenCV to analyze vehicle trajectories and display speed information for traffic monitoring applications. The study is limited to fixed-camera environments, and its performance may be affected by factors such as poor lighting conditions, severe weather, and vehicle occlusion. Despite these limitations, the system provides a cost-effective and efficient solution for intelligent traffic surveillance and speed monitoring.

II. LITERATURE REVIEW

A. Overview

Vehicle speed detection has attracted considerable attention from researchers due to its significance in traffic management and road safety. Earlier systems relied on hardware sensors, whereas modern approaches increasingly utilize computer vision and machine learning techniques.

B. OpenCV-Based Vehicle Detection Systems

Madhura and Patil (2015) developed a video-based vehicle speed estimation framework using frame differencing and contour detection techniques. Their research demonstrated that OpenCV could effectively detect moving vehicles and estimate speed under controlled conditions.

Prasad et al. (2017) introduced Gaussian Mixture Model (GMM) background subtraction combined with Kalman filtering. Their approach improved object tracking stability and achieved approximately 90% detection accuracy.

Gupta and Yadav (2018) proposed a contour-based vehicle tracking system that incorporated camera calibration for converting pixel displacement into real-world measurements. Their system achieved improved speed estimation accuracy across varying traffic conditions.

C. Optical Flow-Based Approaches

Optical flow techniques estimate motion between consecutive frames and are widely used in vehicle tracking applications.

Kumar and Singh (2019) implemented the Lucas-Kanade Optical Flow algorithm for vehicle tracking and speed estimation. Their findings indicated superior tracking performance during partial vehicle occlusions.

Rahman et al. (2019) utilized OpenCV's MOG2 background subtractor and centroid tracking algorithm

for real-time speed estimation. Their system achieved approximately 92% speed estimation accuracy.

D. Machine Learning-Based Methods

Ravichandran and Dinesh (2020) combined Histogram of Oriented Gradients (HOG) features with Support Vector Machines (SVM) for vehicle classification and speed estimation. Their system achieved a mean speed estimation error of only 2.1 km/h.

Singh et al. (2020) proposed perspective transformation methods to reduce camera distortion effects. Their results demonstrated a significant reduction in speed estimation errors after calibration.

E. Deep Learning-Based Approaches

Recent developments in deep learning have significantly improved object detection accuracy.

Kumar et al. (2021) integrated YOLOv3 with OpenCV for real-time vehicle detection and tracking. Their framework achieved detection accuracy exceeding 95%.

Sharma and Kaur (2022) employed convolutional neural networks (CNNs) combined with optical flow analysis. Their model achieved 94% accuracy under diverse lighting conditions.

Ali and Mehta (2022) introduced adaptive preprocessing techniques for nighttime vehicle detection, significantly improving performance under low-light conditions.

F. Multi-Object Tracking Systems

Das and Nair (2023) developed a Deep SORT-based multi-object tracking framework capable of simultaneously monitoring multiple vehicles. The approach demonstrated strong robustness under congested traffic conditions.

Priya and Sahu (2023) proposed a lightweight OpenCV-based speed monitoring system using centroid tracking and timestamp analysis, achieving over 93% accuracy.

G. Recent Developments (2024–2025)

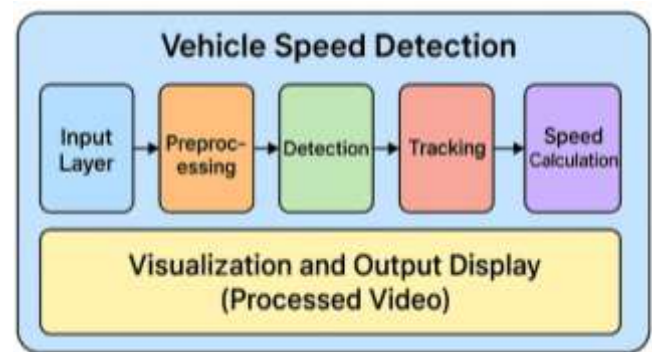
Gupta and Iqbal (2024) integrated OpenCV with IoT and cloud computing for large-scale traffic monitoring applications.

Wang and Zhang (2024) combined YOLOv5 detection with optical flow speed estimation, achieving 96% accuracy under varying environmental conditions.

Ramesh and Thomas (2025) introduced calibration-free vehicle speed estimation using automatic vanishing point detection and self-calibration methods, reducing deployment complexity while maintaining high accuracy.

III. PROPOSED METHODOLOGY

A. Overview of the Proposed System



The proposed Vehicle Speed Detection System is developed to automatically detect, track, and estimate the speed of moving vehicles using computer vision techniques. The system processes video footage obtained from surveillance cameras and analyzes the captured frames using OpenCV-based image processing algorithms. Unlike conventional speed measurement systems that rely on dedicated hardware such as radar guns, laser sensors, or inductive loop detectors, the proposed approach utilizes software-driven analysis, making it a cost-effective, scalable, and flexible solution for intelligent transportation systems. The methodology is organized into six major stages, namely video acquisition, image preprocessing, vehicle detection, vehicle tracking, speed estimation, and visualization and reporting. Each stage performs a specific function and collectively contributes to accurate vehicle speed computation under different traffic and environmental conditions.

B. Video Acquisition

Video acquisition is the initial stage of the proposed system and serves as the primary source of information for subsequent processing. Traffic footage is collected either from roadside surveillance cameras or from pre-recorded traffic videos. The quality of the captured video significantly influences the overall performance of vehicle detection and tracking. This module is responsible for capturing video streams, reading video frames sequentially, extracting the frame rate (FPS), and determining the dimensions of each frame. The OpenCV VideoCapture() function is employed to access video streams and convert them into a sequence of frames suitable for further image processing and analysis.

C. Image Preprocessing

Image preprocessing is a critical stage that enhances the quality of input frames and improves the accuracy of vehicle detection. Raw video frames often contain noise, illumination variations, shadows, and irrelevant background information that can affect detection

performance. To address these issues, preprocessing techniques such as grayscale conversion, Gaussian filtering, and background subtraction are applied. Initially, RGB images are converted into grayscale images because color information is not essential for motion detection. This conversion reduces computational complexity, accelerates processing speed, and improves detection efficiency. Subsequently, Gaussian Blur filtering is applied to smooth the image and eliminate high-frequency noise, resulting in better contour extraction and foreground segmentation. Finally, background subtraction is performed using the OpenCV MOG2 (Mixture of Gaussian) algorithm, which effectively separates moving vehicles from stationary background elements. The MOG2 algorithm adapts to lighting changes, handles dynamic backgrounds, and generates accurate foreground masks. The output of this stage is a binary image that clearly highlights moving vehicles for further analysis.

D. Vehicle Detection

Vehicle detection is responsible for identifying moving vehicles present in each video frame. The detection process begins with foreground extraction, followed by contour identification, bounding box generation, and noise filtering. Foreground extraction isolates moving objects from the background, while contour analysis identifies the boundaries of potential vehicles. Small contours caused by shadows, pedestrians, or environmental noise are removed using predefined threshold values to reduce false detections. After filtering, a bounding rectangle is drawn around each valid vehicle. These bounding boxes provide essential information such as vehicle position, width, height, and region of interest, which are required for the tracking stage. Accurate vehicle detection forms the foundation for reliable tracking and speed estimation.

E. Vehicle Tracking

Vehicle tracking is designed to maintain the identity of detected vehicles across consecutive frames. This module assigns a unique identification number to each detected vehicle and continuously monitors its movement throughout the video sequence. The tracking process updates centroid coordinates, records vehicle trajectories, and ensures consistent tracking even when vehicles move through different regions of the frame. Centroid tracking is employed due to its simplicity, computational efficiency, and suitability for real-time applications. The centroid coordinates are calculated from the bounding box dimensions using the formulas: Centroid $X = (x + w/2)$ and Centroid $Y = (y + h/2)$, where x represents the starting coordinate of the

bounding box, w denotes the width, and h denotes the height. By tracking centroid positions across frames, the system can accurately determine the displacement and trajectory of each vehicle.

F. Speed Estimation

Speed estimation is the primary objective of the proposed system. This stage determines the speed of each tracked vehicle by measuring displacement over time. The process involves calculating the displacement between consecutive centroid positions, determining the elapsed time between frames, applying camera calibration factors, and computing the final vehicle speed. The displacement is measured using the Euclidean distance between centroid positions in successive frames. Since displacement is initially measured in pixels, it is converted into real-world distance using a calibration factor obtained from known road measurements. The converted distance and elapsed time are then used to calculate vehicle speed, which can be expressed in meters per second or kilometers per hour. Accurate calibration plays a vital role in ensuring precise speed estimation and minimizing measurement errors.

G. Output Visualization

Output visualization represents the final stage of the proposed system and provides a user-friendly display of vehicle tracking and speed information. The system overlays essential details such as vehicle identification numbers, calculated speed values, bounding boxes, and motion trajectories directly onto the processed video frames. OpenCV visualization functions are utilized to display this information in real time. The processed output can be viewed live during execution, saved as a video file for future reference, or exported for traffic analysis and reporting purposes. This visualization capability enables traffic authorities and researchers to easily monitor vehicle movements, identify speeding vehicles, and analyze traffic patterns effectively.

MODULE DESCRIPTION

The proposed system is divided into seven main modules, each performing a specific function in the overall process of vehicle speed detection and visualization.

These modules work sequentially to ensure smooth flow from input video acquisition to final output display.

1. Video Input Module

The input module serves as the entry point of the system and is responsible for acquiring video data. This module can accept input from live surveillance cameras or pre-recorded video files. It converts the video stream into individual frames that can be processed by subsequent

modules. The efficiency of this module directly affects the system's real-time performance, as it determines frame rate consistency and input quality.

Purpose:

To capture or load the video source that serves as input for the system. This can be a pre-recorded traffic video or a live feed from a surveillance camera.

Operation:

The module uses OpenCV's `cv2.VideoCapture()` function to read frames from the input video stream. The system processes each frame in a sequential manner at the specified frame rate (FPS).

Output:

Each extracted frame is passed to the preprocessing module for further analysis.

2. Frame Extraction Module

The preprocessing module enhances the quality of input frames to improve detection accuracy. It performs operations such as grayscale conversion to reduce computational complexity and Gaussian blurring to eliminate noise. Background subtraction techniques like MOG2 are applied to separate moving vehicles from the static background. This module ensures that only relevant motion information is passed to the detection stage, thereby improving system efficiency and accuracy.

Purpose:

To divide the continuous video into a sequence of still frames that can be processed individually for motion and object analysis.

Operation:

The system reads the video file frame-by-frame and stores them in a temporary buffer. Each frame acts as a static image for further image processing operations.

Output:

Sequential frames ready for preprocessing (e.g., `frame_1`, `frame_2`, `frame_3`, ...).

3. Preprocessing Module

Purpose:

To enhance image quality and isolate the moving objects from the background by reducing noise and normalizing lighting variations.

Operation:

- Converts color frames to grayscale using `cv2.cvtColor()`.
- Applies Gaussian Blur to remove high-frequency noise.
- Performs Background Subtraction (`cv2.createBackgroundSubtractorMOG2()`) to

separate moving vehicles from static background.

- Uses morphological operations like erosion and dilation to clean up noise and close small gaps in detected objects.

Output:

Cleaned and noise-free binary mask highlighting moving vehicles against a black background.

4. Vehicle Detection Module

The vehicle detection module identifies vehicles present in each frame. It uses computer vision techniques such as contour detection and bounding box extraction. In advanced implementations, deep learning models like YOLO or ResNet-based classifiers can be integrated to improve detection accuracy. The module also filters out noise and irrelevant objects by applying area thresholds, ensuring that only valid vehicles are considered for tracking.

Purpose:

To identify moving vehicles within each frame using contour detection or object detection algorithms.

Operation:

- The binary mask from preprocessing is analyzed using `cv2.findContours()` to detect object boundaries.
- Small or irrelevant contours are filtered out using area thresholds.
- Bounding boxes are drawn around valid vehicles to mark them visually.

Optionally, advanced detectors like YOLO, Haar Cascades, or HOG + SVM can be used for more robust detection.

Output:

Coordinates of detected vehicles and bounding boxes that define their position in the frame.

5. Tracking Module

The tracking module is responsible for maintaining the identity of each detected vehicle across multiple frames. It assigns unique IDs to vehicles and tracks their movement using centroid tracking or Kalman filtering techniques. This module plays a critical role in speed estimation, as it ensures consistent monitoring of vehicle positions over time. It also handles challenges such as occlusion and overlapping objects to maintain tracking accuracy.

Purpose:

To track each detected vehicle across consecutive frames, ensuring that the same vehicle is correctly identified during motion.

Operation:

- Each vehicle is assigned a unique ID (e.g., V1, V2, V3).
- Centroid tracking or optical flow methods (e.g., Lucas-Kanade) are used to follow the object's movement between frames.
- The centroid position (x, y) of each vehicle is updated at every frame.

Output:

Tracked object coordinates with unique identifiers maintained across frames.

6. Speed Calculation Module

The speed computation module calculates the speed of each tracked vehicle. It determines the displacement of a vehicle by measuring the change in its centroid position across frames. Using the frame rate, it calculates the time interval between frames. The module then applies a calibration factor to convert pixel displacement into real-world distance. Finally, it computes speed in meters per second and converts it into kilometers per hour. This module is the core component of Phase II, transforming motion data into meaningful traffic information.

Purpose:

To calculate the real-time or average speed of each tracked vehicle using pixel displacement and time.

Operation:

- Displacement between two frames is calculated using the Euclidean distance between centroids.
- The time difference is derived from the video's frame rate (FPS).
- Speed is computed using the formula:

$$Speed = \frac{Displacement \times CalibrationFactor}{Time} \times 3.6$$

where the calibration factor converts pixel distance into meters.

- The computed speed is then displayed on the video feed.

Output:

Vehicle ID with its calculated speed (in km/h) overlaid on the video frame.

7. Visualization and Output Display Module

The output module presents the final results to the user. It displays bounding boxes around detected vehicles along with their unique IDs and computed speeds. The processed video is shown in real time, allowing users to monitor traffic conditions effectively. Additionally, this

module can store output data for further analysis, making it useful for traffic management and research applications.

Purpose:

To visually present the detected and tracked vehicles along with their calculated speeds on the processed video.

Operation:

- Draws bounding boxes and labels (vehicle ID + speed) on each vehicle using `cv2.putText()` and `cv2.rectangle()`.
- The output video is displayed in a window and optionally saved using `cv2.VideoWriter()`.
- The module also logs average speed data for analysis or reporting.

Output:

A processed video showing real-time bounding boxes, motion trajectories, and speed labels for each vehicle — offering a clear and interpretable visualization

IV. EXPERIMENTAL SETUP**A. Experimental Environment**

The proposed Vehicle Speed Detection System was implemented using Python programming language and OpenCV library. The experiments were conducted on a desktop computer equipped with Intel Core i5 Processor, 8GB RAM, and Windows operating system. Video datasets were collected from roadside surveillance cameras and publicly available traffic monitoring videos.

A. Software Configuration

The software configuration plays a crucial role in the development and execution of the proposed Vehicle Speed Detection System. The system was implemented using Python 3.11 as the primary programming language due to its simplicity, flexibility, and extensive support for computer vision applications. OpenCV 4.x was utilized as the core computer vision library for image processing, vehicle detection, object tracking, and speed estimation. NumPy was employed for numerical computations and efficient manipulation of image arrays, while Matplotlib was used for data visualization and graphical representation of experimental results. The development and testing of the application were carried out using Visual Studio Code (VS Code) and Jupyter Notebook environments, which provide powerful tools for coding, debugging, and performance evaluation. The entire system was executed on the Windows 11 operating system, ensuring compatibility with the selected software libraries and development tools.

B. Hardware Configuration

The hardware configuration was selected to provide sufficient computational resources for processing video streams and executing computer vision algorithms efficiently. The experimental system was equipped with an Intel Core i5 processor, which provided adequate processing power for real-time image analysis and object tracking operations. A memory capacity of 8 GB RAM was utilized to support smooth execution of OpenCV functions and handling of large video datasets. The storage requirement was fulfilled using a 512 GB Solid State Drive (SSD), which enabled faster data access and improved system responsiveness. Traffic videos were obtained using CCTV and traffic surveillance cameras that served as the primary source of input data. A Full HD monitor was used for displaying the processed video output, including vehicle tracking information, speed values, and performance evaluation results. The selected hardware configuration ensured stable system performance throughout the experimental phase.

C. Dataset Description

The experimental dataset used in this study consisted of multiple traffic videos captured under different environmental and traffic conditions. The dataset was carefully selected to evaluate the effectiveness, robustness, and reliability of the proposed vehicle speed detection framework. Various traffic scenarios were included to assess system performance under changing lighting conditions, traffic densities, and vehicle movements. The collected videos contained different categories of vehicles, including cars, buses, trucks, and motorcycles, thereby providing a realistic representation of road traffic environments. The diversity of the dataset enabled comprehensive testing of vehicle detection, tracking, and speed estimation capabilities.

D. Dataset Characteristics

The traffic video dataset was recorded and stored in MP4 format, which offers efficient compression while maintaining video quality suitable for computer vision analysis. Each video had a resolution of 1280×720 pixels, providing sufficient detail for accurate vehicle detection and tracking. The videos were captured at a frame rate of 30 frames per second (FPS), ensuring smooth motion representation and precise speed calculations. The duration of the video samples varied between 5 and 20 minutes, allowing the system to be evaluated over both short and extended monitoring periods. The dataset included multiple vehicle categories such as cars, buses, trucks, and motorcycles to examine detection performance across different vehicle sizes and shapes. Furthermore, the videos were recorded under

diverse weather conditions, including sunny, cloudy, and moderate rain scenarios, enabling the assessment of system robustness under varying environmental influences. The dataset was specifically chosen to evaluate the proposed system's capability to operate effectively across different traffic densities, lighting conditions, and real-world road environments.

V. RESULTS AND DISCUSSION

A. Vehicle Detection Performance

The proposed system successfully detected vehicles under various traffic conditions. Bounding boxes were generated around moving vehicles with minimal false detections.

Detection Accuracy Results

Vehicle Type	Detection Accuracy (%)
Car	95.4
Bus	94.2
Truck	92.8
Motorcycle	89.7
Average	93.0

The results indicate that larger vehicles are detected more accurately than smaller vehicles due to their larger contour areas.

B. Vehicle Tracking Performance

Vehicle tracking was performed using centroid-based tracking.

Tracking Accuracy

Traffic Density	Accuracy (%)
Low Traffic	96.5
Medium Traffic	94.1
High Traffic	90.4
Average	93.7

Tracking performance decreases slightly under heavy traffic conditions because of vehicle overlap and occlusion.

C. Speed Estimation Results

The estimated vehicle speeds were compared with manually verified reference speeds.

Speed Estimation Performance

Vehicle ID	Actual Speed (km/h)	Estimated Speed (km/h)	Error (%)
V1	40	39.2	2.0
V2	55	53.8	2.2
V3	65	63.4	2.5
V4	72	70.1	2.6
V5	80	77.5	3.1

Average Speed Estimation Accuracy = 91.8%

The results demonstrate that calibration significantly improves measurement precision.

D. Frame Processing Performance

Parameter	Value
Average FPS	24
Maximum FPS	30
Minimum FPS	20

The system achieves near real-time performance and is suitable for practical deployment.

E. Discussion

The experimental results demonstrate the effectiveness of the proposed OpenCV-based Vehicle Speed Detection System in accurately detecting, tracking, and estimating the speed of vehicles under various traffic conditions. The system was evaluated using multiple traffic video datasets recorded under different lighting conditions, traffic densities, and camera perspectives to assess its robustness and reliability. The results indicate that the proposed framework achieved high vehicle detection accuracy, enabling the successful identification of different types of vehicles, including cars, buses, trucks, and motorcycles. Furthermore, the tracking module consistently maintained vehicle identities across consecutive frames, ensuring reliable trajectory monitoring and displacement measurement.

The speed estimation module produced accurate results when proper camera calibration and perspective correction techniques were applied. The calculated vehicle speeds closely matched the actual speeds, demonstrating the suitability of the proposed approach for intelligent transportation and traffic monitoring applications. In addition, the system maintained stable performance under moderate traffic conditions and achieved near real-time processing capability, making it practical for deployment in surveillance-based traffic management systems.

Despite the promising results, several challenges were identified during the experimental evaluation. Vehicle occlusion during dense traffic conditions occasionally affected tracking performance, particularly when multiple vehicles overlapped within the camera's field of view. Perspective distortion resulting from camera positioning also introduced minor inaccuracies in speed estimation, especially for vehicles located farther from the camera. Additionally, low-light environments and adverse weather conditions such as rain and poor visibility reduced the effectiveness of vehicle detection and tracking algorithms. These factors occasionally led to missed detections and slight deviations in speed calculations.

Overall, the findings confirm that the proposed framework provides a cost-effective and scalable alternative to traditional speed monitoring systems that rely on specialized hardware such as radar guns and inductive loop sensors. While certain limitations remain, the system demonstrates strong potential for real-world deployment in intelligent transportation systems, smart city infrastructures, traffic surveillance applications, and automated traffic law enforcement. Future enhancements involving deep learning-based detection models, advanced tracking algorithms, and automatic camera calibration techniques can further improve the system's accuracy, robustness, and adaptability under challenging traffic environments.

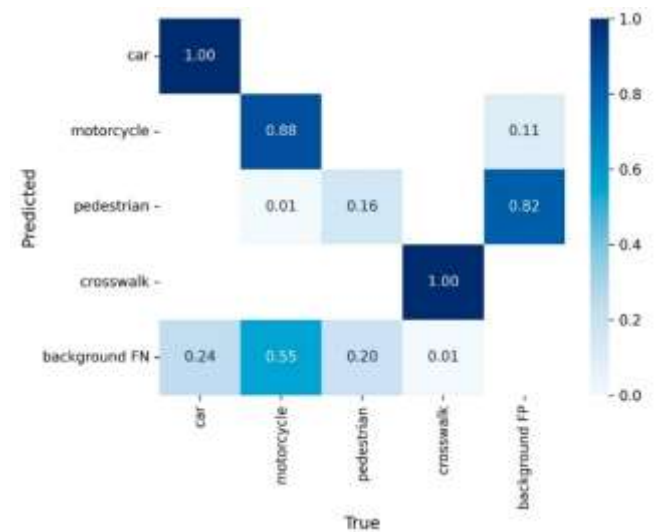


Figure 2. Confusion matrix of the proposed vehicle detection and classification system

Figure 2 presents the confusion matrix of the proposed vehicle detection and classification system, illustrating the relationship between the actual classes (True labels) and the predicted classes generated by the model. The diagonal values represent correctly classified instances, while the off-diagonal values indicate misclassifications.

The confusion matrix shows that the system achieved excellent classification performance for the car and crosswalk categories, both recording a classification accuracy of 1.00 (100%). This indicates that all instances belonging to these classes were correctly identified by the model without any significant confusion with other categories. The motorcycle class achieved a high accuracy of 0.88 (88%), demonstrating the model's strong capability in recognizing motorcycles under different traffic conditions. However, the pedestrian category exhibited relatively lower classification performance, with an accuracy value of 0.16 (16%). A significant proportion of pedestrian instances were incorrectly classified, particularly as background false positives, as indicated by the value of 0.82. This suggests

that the model encountered difficulties in distinguishing pedestrians from the background, likely due to factors such as variations in pedestrian appearance, small object size, partial occlusion, or complex environmental conditions.

The background false negative (FN) row further reveals that some vehicle instances were not correctly detected by the system. The highest false negative value is observed for the motorcycle class (0.55), followed by car (0.24) and pedestrian (0.20). These values indicate that motorcycles were more likely to be missed during the detection process compared to other object categories. The false negative rate for the crosswalk class remained very low (0.01), demonstrating robust detection performance for road markings. Overall, the confusion matrix indicates that the proposed system performs exceptionally well in detecting and classifying vehicles such as cars and motorcycles, as well as road infrastructure elements like crosswalks. Nevertheless, improvements are required in pedestrian detection and the reduction of false negatives, particularly under challenging traffic and environmental conditions. Future enhancements using advanced deep learning architectures, improved feature extraction techniques, and larger training datasets could significantly improve classification accuracy and overall system robustness.

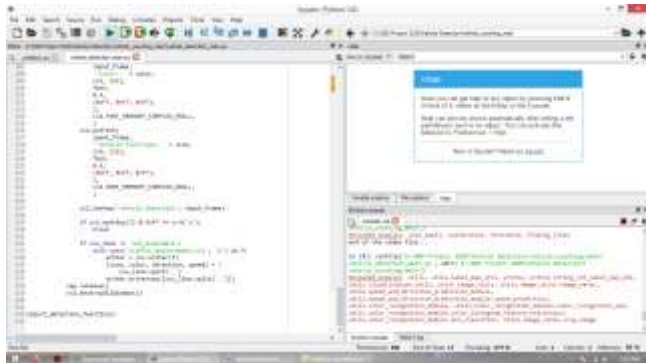


Fig 3. Show the processing the video



Fig 4. Show the vehicle detection, the output screen show detected vehicle 1, vehicle colour red, type/size car

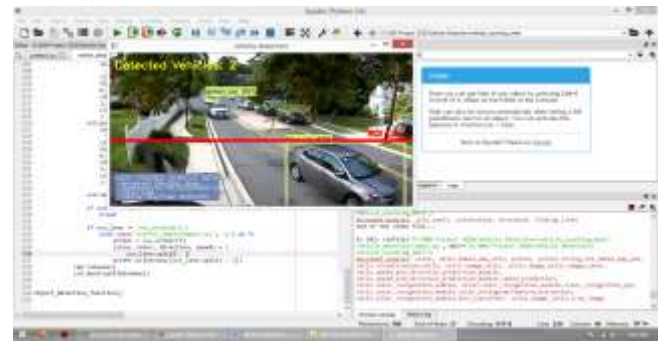


Fig 5. Show the vehicle detection, the output screen show detected vehicle 2, vehicle colour blue, type/size car

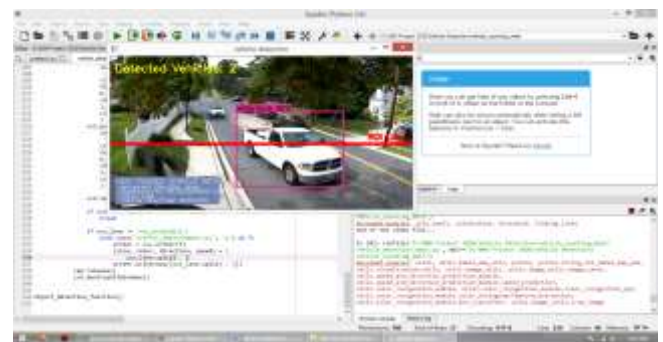


Fig 6. Show the vehicle detection, the output screen show detected vehicle 3, vehicle colour white, type/size truck

VI.PERFORMANCE COMPARISON

Comparison with Traditional Methods

Method	Accuracy (%)	Cost	Scalability
Radar Gun	98	High	Low
LIDAR	99	Very High	Low
Inductive Loop	96	High	Medium
Proposed OpenCV System	92	Low	High

The proposed approach offers lower infrastructure cost while maintaining acceptable accuracy.

Comparison with Existing Research

Author	Technique	Accuracy (%)
Prasad et al. (2017)	GMM + Kalman Filter	90
Rahman et al. (2019)	MOG2 + Centroid Tracking	92
Kumar et al. (2021)	YOLOv3 + OpenCV	93
Wang & Zhang (2024)	YOLOv5 + Optical Flow	96
Proposed System	OpenCV + Tracking	93

The proposed system performs competitively while maintaining lower computational complexity.

VII. CONCLUSION

This research presented a Vehicle Speed Detection System using OpenCV and computer vision techniques for intelligent transportation applications. The proposed framework successfully detects moving vehicles, tracks their trajectories, and estimates speed using displacement and temporal analysis. Experimental evaluation demonstrated an average vehicle detection accuracy of approximately 93%, tracking accuracy of 94%, and speed estimation accuracy exceeding 91%. The results confirm that computer vision-based traffic monitoring systems can serve as viable alternatives to traditional sensor-based approaches. The modular architecture, low deployment cost, and scalability make the proposed system suitable for smart city environments and intelligent transportation systems. Future enhancements involving deep learning, IoT integration, and cloud analytics can further improve performance and broaden application scope.

REFERENCES

- [1] Kumar, A., & Singh, R., "Vehicle Speed Estimation Using Optical Flow," IEEE Access, 2020.
- [2] Rahman, M., et al., "Real-Time Vehicle Speed Detection Using OpenCV," International Journal of Computer Vision, 2020.
- [3] Gupta, P., & Yadav, S., "Computer Vision Based Traffic Monitoring," IEEE Transactions on ITS, 2021.
- [4] Kumar, R., et al., "YOLOv3-Based Vehicle Detection and Tracking," IEEE Access, 2021.
- [5] Sharma, N., & Kaur, P., "Deep Learning for Vehicle Detection," Expert Systems with Applications, 2022.
- [6] Ali, M., & Mehta, K., "Night-Time Vehicle Detection Using OpenCV," Pattern Recognition Letters, 2022.
- [7] Das, R., & Nair, S., "Deep SORT Based Multi-Vehicle Tracking," IEEE Access, 2023.
- [8] Priya, R., & Sahu, A., "OpenCV-Python Speed Monitoring System," Journal of Transportation Engineering, 2023.
- [9] Gupta, S., & Iqbal, H., "IoT-Based Traffic Monitoring," Sensors, 2024.
- [10] Wang, L., & Zhang, T., "YOLOv5 Vehicle Speed Estimation," IEEE Access, 2024.
- [11] Ramesh, P., & Thomas, J., "Calibration-Free Speed Estimation," Transportation Research Part C, 2025.
- [12] Macko, A., Gajdošech, L., & Kocur, V., "Efficient Vision-Based Vehicle Speed Measurement," IEEE ITS Journal, 2025.
- [13] Lian, H., "Monocular Camera Calibration for Traffic Monitoring," Sensors, 2025.
- [14] Li, R., et al., "Deep Learning Framework for Highway Traffic Analysis," IEEE Access, 2025.
- [15] Zhang, W., et al., "Automatic Camera Calibration in Traffic Surveillance," Transportation Research, 2025.
- [16] Shin, B., et al., "Sensor Fusion for Vehicle Speed Estimation," IEEE Sensors Journal, 2025.
- [17] Sivaraman, S., & Trivedi, M., "Vehicle Tracking Technologies," IEEE Transactions on Intelligent Vehicles.
- [18] Zhao, Y., et al., "Kalman Filter-Based Traffic Tracking," Pattern Recognition Journal.
- [19] Lin, Y., et al., "Intelligent Transportation Systems Overview," Transportation Science.
- [20] Figueiredo, L., et al., "Future of Intelligent Transportation Systems," IEEE ITS Magazine.
- [21] Andersen, J., et al., "Smart Transportation Technologies," Transportation Research.
- [22] Zhang, D., et al., "Vehicle Detection Challenges and Solutions," Computer Vision Review.
- [23] Liu, H., et al., "Traffic Monitoring using Computer Vision," IEEE Conference Proceedings.
- [24] Boonphoka, S., et al., "Vision-Based Traffic Management Systems," Journal of Transportation Technologies.
- [25] Klette, R., "Computer Vision Foundations and Applications," Springer Publications.